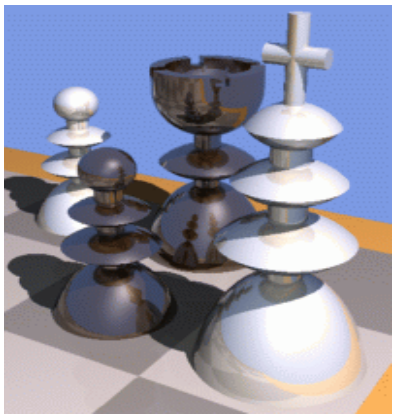
11. Ray-Tracing

Ray heißt Strahl und *to trace* heißt eine Spur verfolgen. Ray-Tracing ist also das „Verfolgen von Strahlspuren“, wobei Lichtstrahlen in verkehrter Richtung durchlaufen werden. Aufbauend auf dem Grundprinzip von Ray-Casting ist Ray-Tracing eine sehr mächtige Methode, mit der neben der korrekten Sichtbarkeit einige wichtige optische Effekte simuliert werden können: Schattierung, Schatten, Spiegelbilder, Lichtbrechung. Die Einfachheit des Verfahrens erlaubt es, auch sehr komplexe Objekte auf diese Art darzustellen, wie Freiformflächen, fraktale Oberflächen, mathematische Funktionen aller Art usw.



Das Ray-Tracing Prinzip

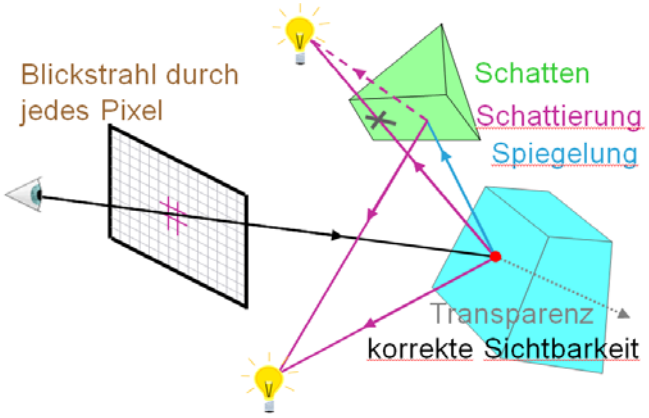
Die Basisidee ist es, das Licht, das auf einen Bildpunkt trifft, in umgekehrter Richtung zu verfolgen (also zu untersuchen, wo es herkommt) und daraus auf das Aussehen dieses Bildpunktes (Pixels) zu schließen.

Korrekte Sichtbarkeit und Schattierung

Wie beim Ray-Casting legt man durch jeden Bildpunkt einen *Blickstrahl* („Primärstrahl“) und schneidet diesen mit *allen* Oberflächen der Szene. Aus den erhaltenen Schnittpunkten wählt man denjenigen aus, der am nächsten zum Bild liegt, und wählt die Schattierung dieses Objektpunktes (aus der Blickrichtung betrachtet) als Wert für den Bildpunkt. Tut man dies für alle Bildpunkte (also im einfachsten Fall für alle Pixel), so erhält man eine Abbildung der Szene in korrekter Sichtbarkeit. Dabei kann ein beliebiges Schattierungsmodell verwendet werden, also z.B. das Phong-Modell.

Schatten

Um die Schattierung eines Punktes zu berechnen braucht man neben der Oberflächennormale auch die Richtungen zu allen Lichtquellen. Eine Lichtquelle hat aber nur dann einen direkten Einfluss auf die Schattierung eines Punktes, wenn der Lichteinfall nicht durch andere Objekte verdeckt ist, wenn also keine Objekte zwischen dem Punkt und der Lichtquelle liegen. Um dies festzustellen, legt man einen *Schattenfühler* (das ist ein „Sekundärstrahl“) vom zu schattierenden Punkt zur Position der Lichtquelle, schneidet diese Gerade mit *allen* Objekten der Szene und berücksichtigt die Lichtquelle nicht, wenn man einen Schnittpunkt zwischen Objekt und Lichtquelle erhält. Auf diese Art erhalten alle Objektteile, die im Schatten eines verdeckenden anderen Objektes bezüglich der Lichtquelle liegen, weniger Lichtquelleneinfluss als Objektteile, die von der Lichtquelle aus ungehindert sichtbar sind, und es entsteht daher automatisch ein Schattenwurf durch die dazwischen liegenden Objekte.

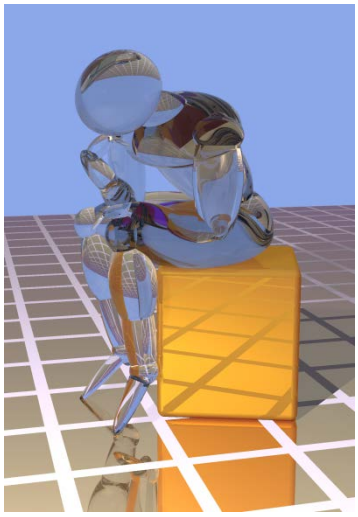


Spiegelbilder

Trifft der Blickstrahl auf ein Objekt auf, das ein Spiegel ist, so sieht man nicht dieses Objekt selbst, sondern jenes Objekt, das vom Auftreffpunkt aus in Spiegelungsrichtung sichtbar ist. Da das Reflexionsgesetz (Einfallswinkel ist gleich Ausfallswinkel) symmetrisch ist, kann man dieses gespiegelte Objekt dadurch finden, dass man den Blickstrahl an der Oberfläche spiegelt und in diese Spiegelungsrichtung einen *Reflexionsstrahl* (das ist auch ein „Sekundärstrahl“) verfolgt, d.h. wieder mit *allen* Objekten schneidet und den zunächst liegenden Schnittpunkt auswählt. Die Schattierung dieses weiteren Auftreffpunktes (betrachtet aus der Eintreffrichtung des Reflexionsstrahles) ist dann das, was der ursprüngliche Blickstrahl sieht. Man beachte, dass das Reflexionsverhalten ganz lokal berechnet wird, man also ohne Mehraufwand gekrümmte Spiegel erzeugen kann.

Transparenz

Es bereitet nun auch keine Schwierigkeiten, transparente Objekte zu behandeln. Ist der erste Auftreffpunkt eines Blickstrahles auf einem durchsichtigen Objekt, so sieht man aus Blickrichtung das, was der durch das durchsichtige Objekt verlaufende *Transparenzstrahl* (auch wieder ein „Sekundärstrahl“) trifft. Dabei ist es ein Leichtes, vermittels des Brechungsgesetzes die Richtung des Transparenzstrahles so zu legen, dass das durchsichtige Material das Licht bricht. Der Transparenzstrahl wird ebenfalls mit *allen* Objekten geschnitten und der zunächst liegende Schnittpunkt



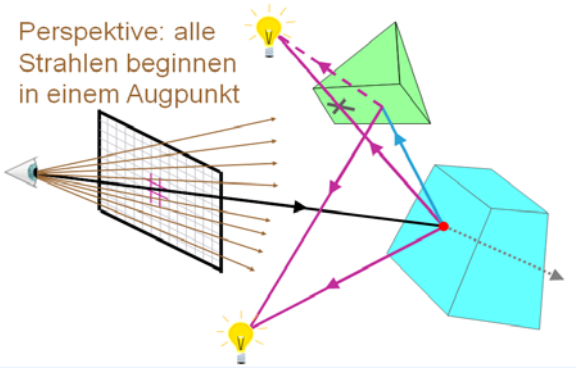
ausgewählt. Die Schattierung dieses weiteren Auftreffpunktes (betrachtet aus der Eintreffrichtung des Transparenzstrahles) ist dann das, was der ursprüngliche Blickstrahl sieht.

Rekursion

Jeder Strahl außer einem Schattenfühler, also jede Gerade, die einen verkehrt durchlaufenen Lichtstrahl repräsentiert, ist grundsätzlich gleichwertig. D.h. trifft so ein Strahl auf eine Oberfläche, so ist es für die Aktion an dieser Stelle unerheblich, ob es sich um einen Primär- oder Sekundärstrahl handelt. Auf diese Weise werden auch Mehrfachspiegelungen und Spiegelungen hinter transparentem Material usw. genauso einfach möglich.

Perspektive

Die Erzeugungsweise der primären Blickstrahlen bestimmt, wie die Abbildung der Szene auf die Bildebene erfolgt. Verwendet man parallele Strahlen, die normal auf die Bildebene stehen, so erhält man eine orthonormale Parallelprojektion der Objekte. Lässt man alle Blickstrahlen von einem fiktiven Augpunkt ausgehen, so wird das Bild in Perspektive erzeugt. Man beachte, dass die Perspektive dabei auf natürliche Weise ohne Mehraufwand entsteht (wenn man von Optimierungsverfahren absieht, die die Parallelität der Strahlen ausnützen).

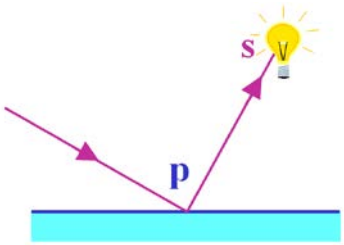


Ray-Tracing Implementierung

Einen Ray-Tracer zu schreiben ist also ganz einfach. Man braucht eine Funktion, die eine Gerade mit allen Objekten schneidet und den vordersten Schnittpunkt zurückliefert.

Ray-Tracing Pseudocode:

```
FOR alle Pixel p₀ DO
 1. lege Blickstrahl vom Auge e aus durch p₀
      schneide mit allen Objekten und wähle den nähesten Schnittpunkt p
 2.FOR alle Lichtquellen s DO
      schneide Schattenfühler p -> s mit allen Objekten
      IF kein Schnittpunkt zwischen p, s THEN Schattierung += Einfluss von s
 3.IF Oberfläche von p ist spiegelnd
   THEN verfolge Sekundärstrahl; Schattierung += Einfluss der Reflexion
 4.IF Oberfläche von p ist transparent
   THEN verfolge Sekundärstrahl; Schattierung += Einfluss d. Transparenz
```

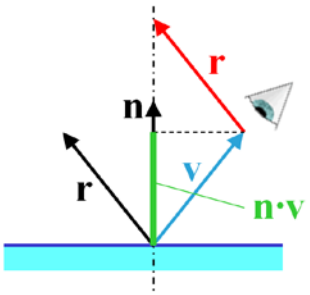


Das Viewing-Koordinatensystem wird normal so aufgestellt, dass die xy-Ebene die Bildebene ist und die Hauptblickrichtung mit der negativen z-Achse zusammenfällt. Strahlen werden in einer parametrisierten Form verwendet:
Startpunkt $\mathbf{p}_0$ plus Parameter t mal Richtungsvektor **d**: $\mathbf{p}(t) = \mathbf{p}_0 + t \cdot \mathbf{d}$.

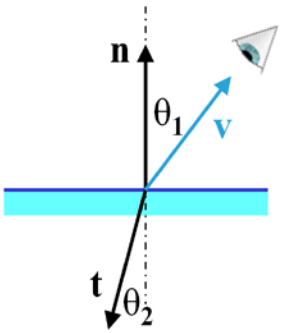
Primärstrahlen sind also **Augpunkt e + t·(Pixel p_0 – Augpunkt e)**,

Schattenfühler sind **Oberflächenpunkt p + t·(Lichtquellenposition s – p)**,

Reflexionsstrahlen sind $\mathbf{p} + t \cdot \mathbf{r}$, wobei $\mathbf{r} = (2\mathbf{n} \cdot \mathbf{v})\mathbf{n} - \mathbf{v}$ die Reflexionsrichtung des Blickstrahles **v** ist, die sich aus dem Reflexionsgesetz Einfallswinkel = Ausfallswinkel ergibt. Diese Berechnung garantiert auch, dass **r** die Länge 1 hat. (siehe rechte Skizze)



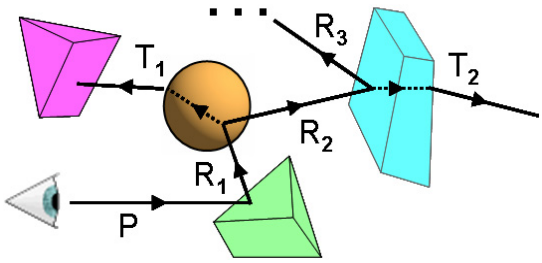
Transparenzstrahlen sind $\mathbf{p} + t \cdot \mathbf{t}$, wobei **t** sich aus dem Snellius'schen Brechungsgesetz $\sin\theta_1 : \sin\theta_2 = \eta_2 : \eta_1$ berechnet (η_i ist der Brechungsindex des Materials i):

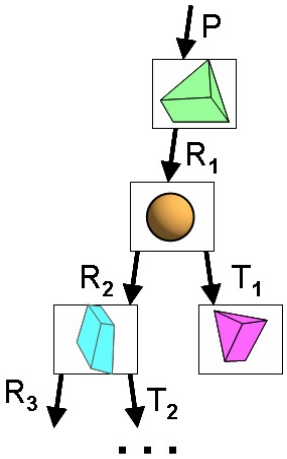


$$\mathbf{t} = -\frac{\eta_1}{\eta_2}\mathbf{v} - (\cos\theta_2 - \frac{\eta_1}{\eta_2}\cos\theta_1)\mathbf{n}$$

Auch der Vektor **t** hat wieder die Länge 1 (siehe linke Skizze).

Wenn man nun solcherart die Lichtstrahlen in verkehrter Richtung „verfolgt“, so entsteht eine rekursive Aufruffolge (siehe Skizze unten), die einem Strahlenbaum entspricht (rechte Skizze). Normalerweise wird dieser Baum aber nicht in dieser Form gespeichert, sondern ist nur eine symbolische Darstellung der Rekursionsaufruffolge.



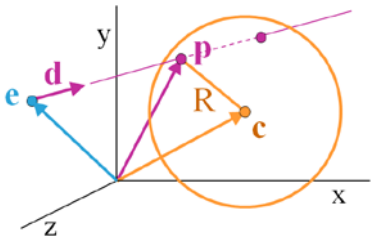


Schnitte zwischen Strahlen und Objekten

Objekte, die mit Ray-Tracing dargestellt werden sollen, müssen wenige Bedingungen erfüllen:
- es muss der Schnittpunkt mit einer Geraden berechenbar sein,
- es muss die Oberflächennormale an dieser Stelle bekannt sein,
- es müssen (an dieser Stelle) Materialeigenschaften vorhanden sein.

Für BReps ist das natürlich einfach, aber ebenso für CSG-Bäume, die rekursiv evaluiert werden. Aber auch viele andere Datenformate erfüllen diese Bedingungen (z.B. Freiformflächen). Für jede Primitivart müssen also Funktionen bereitgestellt werden, die den Schnitt mit einem Strahl berechnen. Als Beispiel wird dies für eine Kugel und für ein Polygon näher beschrieben.

Schnitt Strahl-Kugel



Kugelgleichung: $|\mathbf{p} - \mathbf{c}|^2 - R^2 = 0$

In diese setzt man den Strahl ein: $|(\mathbf{e} + t\mathbf{d}) - \mathbf{c}|^2 - R^2 = 0$

Dann wird zur besseren Lesbarkeit **Δp** eingeführt: $\Delta\mathbf{p} = \mathbf{c} - \mathbf{e}$

Und man erhält eine quadratische Gleichung in t: $t^2 - 2(\mathbf{d}\cdot\Delta\mathbf{p})t + (|\Delta\mathbf{p}|^2 - R^2) = 0$ $\quad (\mathbf{d}^2 = 1)$

Die 2 Lösungen entsprechen den beiden Schnittpunkten mit der Kugel: $t = \mathbf{d}\cdot\Delta\mathbf{p} \pm \sqrt{(\mathbf{d}\cdot\Delta\mathbf{p})^2 - |\Delta\mathbf{p}|^2 + R^2}$

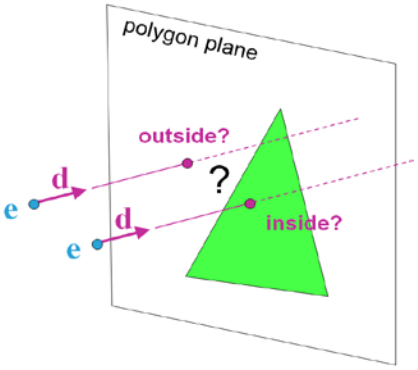
In Fällen, wo $R^2 << |\Delta\mathbf{p}|^2$ ist (das ist durchaus häufig), entstehen in dieser Formel Rundungsfehler. Um diese zu vermeiden kann man $\mathbf{d}^2=1$ ausnützen und die Formel umformen, so dass Rundungsfehler unwahrscheinlicher werden:

$$t = \mathbf{d}\cdot\Delta\mathbf{p} \pm \sqrt{R^2 - |\Delta\mathbf{p} - (\mathbf{d}\cdot\Delta\mathbf{p})\mathbf{d}|^2}$$

Schnitt Strahl-Polygon

Beim Schneiden eines Strahls mit einem Polygon wird zuerst getestet, ob das Polygon überhaupt in die richtige Richtung schaut (siehe Backface Detection). Dann wird der Strahl $\mathbf{p} = \mathbf{p}_0 + t\cdot\mathbf{d}$ in die Ebenengleichung Ax + By + Cz + D = 0 des Polygons ein-

gesetzt, die sich, da **n**=(A,B,C) gilt, auch in der Form **n**·**p** = –D anschreiben lässt: **n**·(**p**$_0$ + t·**d**) = –D. Daraus erhält man t = –(D + **n**·**p**$_0$)/**n**·**d** , das in die Strahlgleichung eingesetzt den Schnittpunkt mit der Ebene ergibt.



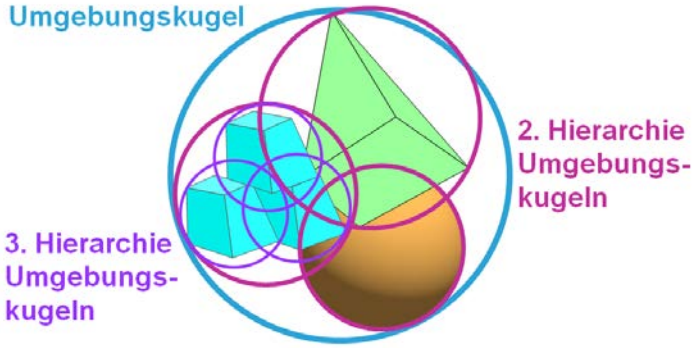
Nun muss noch überprüft werden, ob der Schnittpunkt innerhalb der Polygonkanten liegt, oder neben dem Polygon (siehe Abb. rechts). Dies kann man nach Projektion auf eine Hauptebene in 2D machen.

Ray-Tracing Beschleunigung

Ray-Tracing ist ein extrem aufwändiges Verfahren. Eine Szene mit nur 1000 Polygonen oder Objekten auf eine Fläche von 1000x1000 Pixel abzubilden erfordert ohne Optimierungen alleine für die Primärstahlen bereits 10^9 Schnittberechnungen. Daher ist es notwendig, das Verfahren signifikant zu beschleunigen. Die wichtigste Methode dazu ist es, die Anzahl der notwendigen Schnittberechnungen durch Ausnutzung von Kohärenz zu reduzieren.

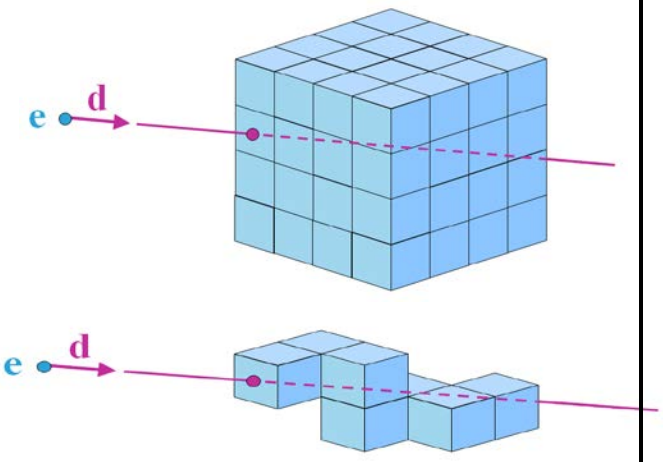
Objektumgebungen

Bevor man alle Einzelteile eines komplexeren Objektes (oder eines Szeneteiles) mit einem Strahl schneidet, kann man überprüfen, ob der Strahl überhaupt in die Nähe dieses Objektes kommt. Dazu werden in der Datenstruktur zu solchen komplexeren Objekten *Umgebungskugeln* (bounding spheres) eingefügt, die das Objekt ganz umschließen. Strahlen, die diese Umgebungskugel nicht treffen, treffen auch das Objekt sicher nicht, und man erspart sich viele unnötige Schnittberechnungen. Das Konzept lässt sich hierarchisch anwenden, d.h. komplexe Teilobjekte erhalten alle ebenfalls Umgebungskugeln, deren Teile wieder usw., bis man an einfache Objekte gelangt. Auf diese Weise verringert man die tatsächlichen Schnittversuche von O(n) auf etwa O(log n). Statt Umgebungskugeln kann man beliebige andere Objektumgebungen verwenden, z.B. Umgebungsquader. Es gilt hier abzuwägen, ob der Mehraufwand durch deren kompliziertere Form den Gewinn durch deren engeres Anliegen rechtfertigt.



Raumteilungs-Methoden

Alternativ kann man auch den ganzen Raum, in dem sich die Szene befindet, unabhängig von den Objekten in ein regelmäßiges Raster unterteilen. Dabei ist es egal, ob man die Teilwürfel in einem Array abspeichert oder einen Octree dazu verwendet. Man braucht dann nur jeweils die Objekte mit dem Strahl zu schneiden, die in den Teilräumen liegen, durch die der Strahl durchgeht. Man braucht also eine schnelle Berechnung, welches der nächste Teilwürfel auf dem Strahlpfad ist. Hat man in einem Teilwürfel einen Schnittpunkt gefunden, kann man aufhören! Dazu bieten sich Algorithmen ähnlich einem 3D-Bresenham-Verfahren an.



Für einzelne Teilwürfel kann man auch so vorgehen:

Der Strahl **p**(t) = **p**$_0$ + t·**d** tritt an **p**$_{in}$ in den Teilwürfel ein. Die Normalvektoren der Würfelflächen sind (1,0,0), (–1,0,0), (0,1,0), (0,–1,0), (0,0,1), (0,0,–1). Für die drei Flächen mit **d**·**n** > 0 (die anderen kommen nicht in Frage!) bestimmt man den Schnittpunkt mit dem Strahl und wählt den vordersten Punkt (kleinstes t) aus. Diese Methode funktioniert auch, wenn die Würfel unterschiedlich groß sind (z.B. in einem Octree).

$$\mathbf{p}_{out,k} = \mathbf{p}_{in} + t_k \mathbf{d}$$

$$\mathbf{n}_k \cdot \mathbf{p}_{out,k} = -D_k$$

$$t_k = \frac{-D_k - \mathbf{n}_k \cdot \mathbf{p}_{in}}{\mathbf{n}_k \cdot \mathbf{d}}$$